Requested Patent: JP6342386A

Title: SYSTEM AND METHOD FOR COMPUTER SYSTEM PROFILING.

Abstracted Patent: EP0501076

Publication Date: 1992-09-02

Inventor(s): KELLER THOMAS WALTER (US); URQUHART ROBERT JOHN (US)

Applicant(s): IBM (US)

Application Number: EP19910311868 19911220

Priority Number(s): US19910662521 19910228

IPC Classification: G06F11/34

Equivalents: JP2777496B2

ABSTRACT:

The invention disclosed herein is a system and method for comprehensive, non-invasive profiling of a processor whereby feedback is provided to a programmer of the execution dynamics of a program. In a preferred embodiment a partial real-time reduction (40) is provided of selected trace events employing the environment's trace facility, and a post-processing function (42) is then performed. A trace hook is provided in the environment's periodic clock routine which captures the address to be returned to following this timer's interrupt, and further captures the address of the caller of the routine represented by the first address. The frequency of occurrences of the first address is collected and correlated (39) to various virtual address spaces and corresponding subroutine offsets within those virtual address spaces. By employing the assembly and source code listing of programs, the address frequencies are then correlated back to specific instructions (49), and from information in the assembly listing accumulated time is further correlated against specific lines of source code. A profile is generated (55) indicating the amount of time spent by the processor in various processes, kernel, shared library, and user spaces, and subroutines correlated to the lines of source code for negligible additional processor run time.

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（Ａ）**　(11)特許出願公開番号

特開平6－342386

(43)公開日　平成６年(1994)12月13日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｇ０６Ｆ　11/34		Ｍ　9290－5Ｂ		
11/30		Ａ　9290－5Ｂ		

審査請求　有　　請求項の数36　ＯＬ　（全 11 頁）

(21)出願番号　特願平3－309469

(22)出願日　平成３年(1991)11月25日

(31)優先権主張番号　６６２５２１

(32)優先日　1991年２月28日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　390009531
インターナショナル・ビジネス・マシーンズ・コーポレイション
ＩＮＴＥＲＮＡＴＩＯＮＡＬ　ＢＵＳＩＮＥＳＳ　ＭＡＳＣＨＩＮＥＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国10504、ニューヨーク州　アーモンク　（番地なし）

(72)発明者　トマス、ウォルター、ケラー
アメリカ合衆国テキサス州、オースチン、サークル、リッジ、ドライブ、1414

(74)代理人　弁理士　頓宮　孝一　（外５名）

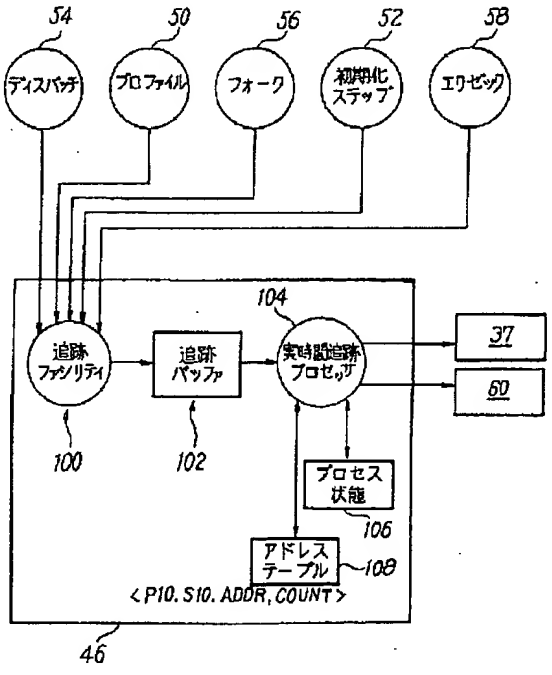
最終頁に続く

(54)【発明の名称】　コンピュータシステムプロファイリングシステム及び方法

(57)【要約】

【目的】　一つのプログラムの実行ダイナミックスをプログラマにフィードバックするようになったプロセッサの総合的非侵入形プロファイリングシステム及び方法の提供。

【構成】　エンバイロメントの追跡ファシリティを用いる選ばれた追跡イベントについて部分的な実時間の減少が得られる。追跡フックがこのエンバイロメントの周期的クロックルーチンに与えられ、このタイマーの割込み後にもどされるべきアドレスを得ると共に第１のアドレスにより表わされるルーチンの呼者のアドレスを得る。



実時間プロファイラ処理

1

【特許請求の範囲】

【請求項1】プロセッサとコードにより発生される命令についてアドレスを記憶してその命令を実行するプログラムカウンタを有するコードを実行するためのコンピュータシステムエンバイロメントにおいて、下記段階を含む、上記エンバイロメント用の上記コードをプロファイルするための方法：上記命令に対応する上記アドレスを上記プログラムカウンタからサンプリングする段階；上記サンプリングされたアドレスの頻度カウントを発生する段階；上記カウントから上記アドレスに対応する上記命令を実行するため上記プロセッサにより使用された時間の表示をとり出す段階。

【請求項2】前記アドレスは前記命令のほぼすべてについてのサンプルである請求項1の方法。

【請求項3】前記表示は前記コードの実行コンポーネントで使用される時間の測定値である請求項1の方法。

【請求項4】前記コンポーネントの夫々は前記コードにより行われる異なるプロセスである請求項3の方法。

【請求項5】前記コンポーネントの夫々は前記コードにより行われる一つのプロセス内の一つのルーチンである請求項3の方法。

【請求項6】前記コンポーネントの夫々は前記コードの原始プログラムステートメントに対応する請求項3の方法。

【請求項7】前記コンポーネントの夫々は前記コードのアッセンブリステートメントに対応する請求項3の方法。

【請求項8】プロセッサとコードにより発生される命令についてアドレスを記憶しその命令を実行するプログラムカウンタを有する、コードを実行するためのコンピュータシステムエンバイロメントにおいて、下記要件を含む、上記エンバイロメントについての上記コードをプロファイリングするためのコンピュータプログラム：上記命令に対応する上記アドレスを上記プログラムカウンタからサンプリングするためのプログラムコード手段；上記サンプリングされたアドレスの頻度カウントを発生するためのプログラムコード手段；上記カウントから、上記アドレスに対応する命令を実行する上記プロセッサにより使用された時間の表示をとり出すプログラムコード手段。

【請求項9】前記アドレスは前記命令のほぼすべてについてのサンプルである請求項8のプログラム。

【請求項10】前記表示は前記コードのコンポーネントの実行に使用された時間の測定値である請求項8のプログラム。

【請求項11】前記コンポーネントの夫々は前記コードにより行われる異なるプロセスである請求項10のプログラム。

【請求項12】前記コンポーネントの夫々は前記コードにより行われる一つのプロセスの一つのルーチンである

2

請求項10のプログラム。

【請求項13】前記コンポーネントの夫々は前記コードの原始プログラムステートメントに対応する請求項10のプログラム。

【請求項14】前記コンポーネントの夫々は前記コードのアッセンブリステートメントに対応する請求項10のプログラム。

【請求項15】複数のプロセスを行うプログラムコードを実行するコンピュータシステムにおいて、下記段階を含む、上記プロセスのすべてをプロファイルするための方法：上記コードの1回の走行時間の実行中上記プロファイリングのためのすべてのデータを得る獲得段階；上記得たデータの関数として上記プロファイリングを発生する段階。

【請求項16】前記獲得段階が下記段階を含む、請求項15の方法：前記コードの前記実行中に前記複数のプロセスの内の異なった一つの夫々について複数の＜ＰＩＤ，ＳＩＤ，ＰＣＮ＞組からなる状態情報を発生する段階。

【請求項17】前記プロファイリングの発生段階は下記段階を含む請求項16の方法：前記システムの対応するアドレスに同一の前記組の累積カウントを相関させる段階。

【請求項18】複数のプロセスを行うプログラムコードを実行するコンピュータシステムにおいて、下記要件を含む、上記プロセスのすべてをプロファイリングするためのシステム：上記コードの1回の走行時間実行中に上記プロファイリングのためのすべてのデータを得るための獲得手段；上記得られたデータの関数として上記プロファイリングを発生するプロファイル発生手段。

【請求項19】前記獲得手段は下記要件を含む、請求項18のシステム：前記コードの前記実行中に前記複数のプロセスの内の異なった一つの夫々について複数の＜ＰＩＤ，ＳＩＤ，ＰＣＮ＞組からなる状態情報を発生する発生手段。

【請求項20】前記プロファイル発生手段は下記要件を含む、請求項19のシステム：前記システムの対応するアドレスに同一の前記組の累積カウントを相関させる手段。

【請求項21】コンピュータエンバイロメント内で実行するマルチプロセスプログラムコードについて、下記段階を含む、走行時間およびポスト処理インターバルを有する上記コードをプロファイリングする方法：上記走行時間インターバルにおいて、上記マルチプロセスのすべてのプロファイリングを示す第1のユーザ特定大域コマンドを発生する段階；上記ポスト処理インターバルにおいて上記マルチプロセスのすべての上記プロファイリングを示す第2のユーザ特定大域コマンドを発生する段階；上記第1および第2大域コマンドに応じて上記マルチプロセスのすべての上記プロファイリングをとり出す

3

段階。

【請求項２２】コンピュータエンバイロメント内で実行するマルチプロセスプログラムコードについて、下記要件を含む、走行時間およびポスト処理インターバルを有する上記コードをプロファイリングするシステム：上記走行時間インターバルにおいて上記マルチプロセスのすべてのプロファイリングを示す第１のユーザ特定大域コマンドを発生する手段；上記ポスト処理インターバルにおいて上記マルチプロセスのすべての上記プロファイリングを示す第２のユーザ特定大域コマンドを発生する手段；上記第１および第２大域コマンドに応じて上記マルチプロセスのすべての上記プロファイリングをとり出す手段。

【請求項２３】マルチプロセスを実行するコンピュータシステムにおいて、下記段階を含む、上記システムで実行するプログラムをプロファイリングする方法：実行されるべきプロファイルの表示を発生する段階；１回の走行時間実行中に上記プロファイルについてのデータを得る段階；その後、実行されるべき詳細サブプロセスレベルプロファイリングの表示を発生する段階；上記得られたデータからポスト処理インターバルにおいて上記詳細サブプロセスレベルプロファイリングを発生する段階。

【請求項２４】アッセンブリリストを発生するために前記コードを再コンパイルする段階を更に含み、
前記詳細サブプロセスレベルプロファイリング段階も上記アッセンブリリストから発生される請求項２３の方法。

【請求項２５】前記サブプロセスプロファイリングは下記プロファイルを含むグループからの少くとも一つのプロファイルを含む請求項２４の方法：ルーチンプロファイル；原始ステートメントプロファイル；アッセンブリステートメントプロファイル。

【請求項２６】マルチプロセスを実行するコンピュータシステムにおいて、下記要件を含む、上記システムで実行するプログラムをプロファイリングするためのシステム：実行されるべきプロファイルの表示を発生する手段；１回の走行時間実行中に上記プロファイル用のデータを得る手段；実行されるべき詳細サブプロセスレベルプロファイリングの表示を発生する手段；上記得られたデータからポスト処理インターバルで上記詳細サブプロセスレベルプロファイリングを発生する手段。

【請求項２７】アッセンブリリストを発生するために前記コードを再コンパイルする手段を更に含み、
前記詳細サブプロセスレベルプロファイリングも上記アッセンブリリストから発生される請求項２６のシステム。

【請求項２８】前記サブプロセスプロファイリングは下記プロファイルの内の少くとも一つのプロファイルを含む請求項２７のシステム：ルーチンプロファイル；原始ステートメントプロファイル；アッセンブリステートメントプロファイル。

4

【請求項２９】プログラムコードを実行するためのコンピュータシステムにおいて、下記段階を含む、上記コードをプロファイリングする方法：上記コードを非侵入型でコンパイルして実行可能コードを発生する段階；上記実行可能コードを走行させる段階；上記実行可能コードの走行中プロファイリングデータを集める段階；上記集められたデータの関数として一つのプロファイリングを発生する段階。

【請求項３０】マルチプロセスを実行するプロセスにおいてオペレーティングシステム核を有するコンピュータシステムにおいて、下記段階を含む、上記マルチプロセスをプロファイリングする際に上記マルチプロセスの実行中の使用方法：上記オペレーティングシステム核の周期的クロックルーチンにおいて追跡フックメカニズムをつくる段階；上記オペレーティングシステム核内に、プロセス識別、名前、および現在走行中のプロセス対応物を初期化し維持するのに充分な付加的追跡フックメカニズムをつくる段階；上記追跡フックおよび付加追跡フックメカニズムに応じて予め定めた時間インターバルで追跡イベントを発生する段階；プロセス状態の変化に応じてその変化の発生時点で追跡イベントを発生する段階；下記を含む上記追跡イベントの関数として追跡バッファをつくり維持する段階：夫々＜現在実行中のプロセス識別、プログラムカウンタ値＞組の夫々固有のインスタンスおよびこの夫々の固有のインスタンスのくり返し回数のカウントに対応する、複数のプログラムカウンタフックデータフィールド；プロセス名、識別と上記プログラムカウンタフックデータフィールドとの間の対応性を維持する複数のフィールド。

【請求項３１】前記ポスト処理インターバルに下記段階を含む、請求項３０の方法：複数のプログラミングコンストラクトに対するプログラムカウンタ頻度カウントの相関を発生する段階。

【請求項３２】前記コンストラクトは下記を含むグループの内の少くとも１つを含む請求項３１の方法：プロセス；スペース識別；ルーチン；コードの原始ライン；アッセンブリ命令。

【請求項３３】前記頻度カウントを発生する段階は上記カウントを記憶するためのカウントメモリスペースを維持する段階を含み、上記スペースは一つのプロファイリングインターバルにおいて生じるカウントに機能的に関係する大きさを有する請求項１の方法。

【請求項３４】前記頻度カウントを発生する手段は上記カウントを記憶するためのカウントメモリスペースを維持する手段を含み、上記スペースは一つのプロファイリングインターバルにおいて生じるカウントに機能的に関係する大きさを有する請求項８のシステム。

【請求項３５】非プロファイリングのための実行についてのエクゼキュータブルを発生するためにコンパイルさ

5

れているプログラムを実行するためのシステムにおいて、下記段階を含む、上記システムをプロファイリングするための方法：上記システムで上記エクゼキュータブルを実行する段階；上記システムでの上記エクゼキュータブルの上記実行中プロファイリングデータを集める段階；上記集められたデータの関数として上記エクゼキュータブルのプロファイリングを発生する段階。

【請求項３６】前記プロファイリングを発生するための上記エクゼキュータブルは前記非プロファイリング用に前記システムで実行するための前記エクゼキュータブルから変更されていない請求項３５の方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明はコンピュータシステムにおけるプロセッサの実行時間をプロファイルするための技術に関し、詳細には追跡駆動プロファイリングに関する。

【０００２】

【従来の技術】種々のコンピュータシステムにより発生されるコードの性能を改善するためにはコードの実行においてプロセッサにより時間を消費されるところを決定する必要がしばしばあり、そのような努力はコンピュータ処理の分野では「ホットスポット」の位置決定として一般に知られている。理想的にはそのようなホットスポットを命令そしてまたはコードレベルのソースラインで分離して注意をコードにとっての改善から有利な領域に向けるようにする。

【０００３】例えばそのようなホットスポットを命令レベルに対し分離することによりコンパイラの記述者は次善の（suboptimal）コード発生の領域を見つけることが出来、それによりそれらの領域でのコード発生効率の改善にその努力を向けることが出来る。命令レベルのディテールの他の潜在的に重要な使用は将来のシステムの設計者に対するガイダンスを与えることである。適正なプロファイリングツールを有するそのような設計者は与えられたレベルのハードウェア技術に使用しうるハードウェアを最適化するために改善を必要とする特性コードシーケンスそしてまたは単一命令を見い出すことが出来る。

【０００４】同様に、ホットスポットをコードレベルのソースラインに分離することでアルゴリズム上の取引きを行うためにアプリケーションの開発者に必要なディテールのレベルが与えられる。プログラムの大きな実行時間を必要とするところについてのプログラマの演算的な予想は多くの理由で誤っていることが多い。まず、プログラマはハードウェアおよびソフトウェアシステムの複合ダイナミックスについての包括的な理解をめったに有していない。第二に、コンパイラ自体はしばしばプログラマの仮定に対応するコードを発生しない。従って、プログラマが容易に理解しうるように一つのプログラムの

6

実行ダイナミックスについてプログラマに情報をフィードバックするシステムを与えることが極めて望ましい。

【０００５】このように、“プロファイリング”と呼ばれる総合的なＣＰＵの使用をモニタするための種々の方法が開発されている。一つの方法は解析中のプログラムに単に命令を付加してそれがそれ自体をアクセスしうるようにするものである。しかしながらこれは望ましくない侵入（invasiveness）の特性を導入することによりプロファイリングのために必要な変更が測定しようとする物のダイナミックスに変化を生じさせる可能性が生じる。他の方法は外部の特殊なハードウェアモニタの開発である。しかしながらこの方法も多くの欠点を含み、その開発に関連するコストと開発を行ってもその可能性の問題がある。

【０００６】場合によってはそのようなプロファイリングの必要性は特に厳しいが、その場合に固有の特性により現存する方向では満足されない。そのようなものの一例はＩＢＭのＡＩＸ™オペレーティングシステムを動作させるコンピュータのＲＩＳＣシステム／６０００™ラインである（ＲＩＳＣ／６０００およびＡＩＸはＩＢＭの商標）。このハードウェアとソフトウェアの詳細はＩＢＭ社の“IBM RISCSystem/6000Technology”，第１版１９９０。ＳＡ２３－２６１９に示されている。

【０００７】そのような場合についてプロファイリングを与える一つの特定の試みがProc.ACM SIGPLAN Symposium on Compiler Construction, １９８２年６月のGraham他“Ｇｐｒｏｆ：A Call Graph Execution Profiler”に示される、“Ｇｐｒｏｆ”と呼ばれるシステムである。このプロファイリングシステムにはいくつかの問題がある。まず共用ライブラリサポートがなく、非共用ライブラリとプログラムのコンパイレーションが必要である。このシステムは多重プロセスの同時プロファイリング用のサポートを与えず、走行しうるすべてのプロセスはルーチンレベルのプロファイリング用に再コンパイルされねばならず、このシステムは侵入的（invasive）（例えばプロファイルされるべき実行可能なコードを変更する）であり、そして付加メモリのプロファイリングにプロファイルされるべきプログラムのスペースのほぼ半分を与えねばならない。更に、プロファイルされるべきプロセスのセットをプロファイリングを与えるために再構成しなければならないことに加えて、ルーチンレベルのプロファイリングを与えることが出来るにすぎず、原始ステートメントまたは命令レベルのプロファイリングを与えることが出来ず、すべてのＣＰＵの使用を要約せず一時に一つのユーザプログラムのそれのみを要約するのであり、更にしばしばその侵入性によりユーザのＣＰＵ時間を大幅に時には３００％近くまで増加させねばならない。

【０００８】このため、他の方法、例えばLarry B Weber の“Compilers Untock RISCSecret”,ESD, １９８９

7

年１２月、pp２６－３２にＭＩＰＳコンピュータシステム社のＰＩＸＩＥシステムを含む場合におけるプロファイリングのための方法が提案されている。

【０００９】このシステムではプロファイルされるべきプロセスの実行可能なオブジェクトが分析されそして“基本ブロック（basic block)”と呼ばれる命令の原始シーケンス（atomic sequence)毎に、基本ブロックの放出されたシーケンスからその基本ブロックの実行の開始をリポートするイベントを出すフックにつづき再構成される。この放出されたイベントシーケンスから各基本ブロックの実行頻度が走行時間中に維持される。サブジェクトポスト処理段階において、この発生頻度はプログラムの原始ステートメントとルーチンに対して相関されて実行時間プロファイルを与える。

【００１０】この方法はプログラムカウンタをサンプリングすることにより得られる予測についての直接的な測定の利点を与えるが、共用ライブラリサポートがなく、多重プロセスのサポートがないという欠点を有し、プログラム実行可能なスペースを３倍まで増加させプログラムのエクゼキュータブル（executables)を１０倍以上にしなければならない。

【００１１】更に他のプロファイリングシステムの開発がなされており、それは ZiyaAral他“non-Intrusive and Interactive Profiling in Parasight”, Proc.ACM/SIGPLAN，１９８８年８月、pp２１－３０に示されている。これにおいては付加的な走行時間を生じさせる侵入性は選択されたコードシーケンスの実行時間を直接測定するように問題のコードシーケンスを選択的に変更しそして走行時間の測定をとらえて処理するために付加的なプロセスを用いることにより減少する。

【００１２】

【発明が解決しようとする課題】以上から、上記の場合をサポートするためのプロファイラ技術は多くの改善を必要とすることは明らかである。特に、多重プロセスおよび多重ユーザエンバイロメント、共用ライブラリ（動的にロードされる共用オブジェクト）、核（kernel）およびユーザ実行スペース、および核拡張（核に動的にロードされる拡張）をサポートしうるプロファイラが必要とされる。

【００１３】プロファイラにおいて特に望ましく大いに必要とされる要件は便利さと非侵入性（non-invasiveness）の特性に関している。これら二つの因子は強く関係すると共にそれら自体に利点を有する。

【００１４】便利さの点ではユーザが特別の手順、再構成、再リンキングまたは再構築を必要とせずに現存する走行コードを非常に容易にプロファイルすることが出来るようなプロファイリングツールを与えることが極めて望ましい。更に、同じく非侵入性のプロファイリングツールを与えることが極めて望ましい。この総合的な特徴は各プロセス、核、核拡張、ユーザおよび共用オブジェクトについてのすべてのプロセスとすべてのアドレスドメインのプロファイルを与える。非侵入性の特徴はエクゼキュータブルとサポートエンバイロメントがプロファイリングであるかないかにかかわらず仮想的に同一であり、この特性を得るために特別の努力を必要としないことである。従来のシステムは例えばしばしばＣＰＵとメモリの利用を過度のものとするような命令レベルでのプロファイルのためにエクゼキュータブルの変更を必要とする。非侵入性の重要性は集められた統計量が歪みを受けず、すべての命令ストリームと参照されたアドレスが維持されることである。後者は特にＴＬＢ、データおよび命令キャッシュ、レジスタ、メモリのようなハードウェア装置の過使用に関係するパフォーマンスの問題をさがすとき重要である。

【００１５】以上の理由でエンバイロメントのユーザプログラムの実行中（並びにＣＰＵがアイドル中の時間部分）のすべてのプログラム（プロセス）走行、核を含む、すべてのユーザの過度のＣＰＵの使用をリポートすることが出来、それによりユーザが大域センスでＣＰＵの使用を決定しうるようなプロファイリングツールが極めて望ましい。そのようなプロファイラは、プログラマがＣＰＵにより最も使用されるプログラムの部分を知るに有効なＣＰＵバウンドであるプログラムを調査するためのツールとしても望ましい。更に、特別なコンパイラフラグあるいはリンカーオプションをコンパイルすることなく実行可能なプログラムを用いて走行出来、それによりサブプログラムプロファイルがすでに構成されている任意の実行可能なモジュールから得られるようになったプロファイラが極めて望ましい。

【００１６】

【課題を解決するための手段】本発明はプロセッサの総合的な非侵入プロファイリングのためのシステムおよび方法であり、プログラムの実行ダイナミックスのプログラマにフィードバックを与えるものである。好適な実施例では部分実時間の減少が、エンバイロメントの追跡ファシリティを用いる選ばれた追跡イベントについて与えられ、そしてポスト処理機能が行われる。追跡フックがエンバイロメントの周期的クロックルーチンに与えられる。このルーチンはこのタイマの割込み後にもどされるべきアドレスをとらえ、そして更に第１アドレスで表わされるルーチンの呼者のアドレスもとえらる。

【００１７】第１アドレスの発生頻度が集められて仮想アドレススペースおよびその内の対応するサブルーチンオフセットに相関される。プログラムをリストするアッセンブリと原始コードを用いることにより、アドレス頻度が特定の命令に相関されそしてアッセンブリリスト内の情報から累積時間が特定の原始コードラインに対し相関される。一つのプロファイルが、種々のプロセス、核、共用ライブラリ、ユーザスペースおよび無視しうる付加プロセッサ走行時間について原始コードラインに相

9

関されるサブルーチンにおいてプロセッサが消費する時間の量を示すものとして発生される。

【0018】

【実施例】図1～5によりこのプロファイリングプロセスの詳細を述べる。続いて図6によりそのプロファイリングに適した代表的なコンピュータエンバイロメントを説明する。プロファイリングの説明に関して、まず図1、2によりその出力の高レベルの説明を行い、続いて図3～5により本発明の動作の詳細を述べる。

【0019】図1において、本発明により発生される全体のプロファイルが概略的に示されている。コラム11のようなコラムは多重プロセス計算エンバイロメントで実行されうる種々のプロセスに対応する。各プロセスについてこのプロファイラは位置13に生じるような総合カウントの目安を発生する。このカウントは特定のプロセス11の実行中に生じる周期的サンプルから集められた総合カウントに対応し、そのプロセスの実行における総合CPU実行時間を表わす。

【0020】図1において、コラム11のような与えられたプロセスすなわちコラムについて総合カウント13は、プロセッサがユーザ、共用または核メモリアドレススペース（ここでは単にスペースという）で実行中に生じたものに更に副分割される。複数の列は図1では「ユーザ」、「共用」、「核」で示す代表的なプロファイルに示される。このように、例えばボックス15に生じるカウントはプロセス11が共用スペースで実行中に生じたカウントに対応し、ボックス19に生じる総合カウントはプロセス17が核、共用およびユーザスペースで実行されたときに生じるものに対応する。

【0021】図2は本発明によりプロファイリングするに適した、代表的なマルチプロセス、多スペース、マルチユーザ計算エンバイロメントを示す。用語「プロファイル」および「プロファイリング」は夫々「実行時間プロファイルリポート」および「実行時間プロファイリング」を省略したものである。

【0022】図2はマルチプロセス、マルチスペース計算エンバイロメントと本発明により与えられるプロファイリングの種々の能力との間の関係を示す。すなわち、マルチプロセス、マルチスペースエンバイロメントは10で概念的に示してある。図1と同様にプロセス18のような特定のプロセスのユーザスペースが12で示してある。同様に、プロセス18についてボックス14は各ユーザスペースによりアクセス可能な共用スペースを示す。プロセス18について、ボックス16はシステム呼出しにより各ユーザスペースにアクセス可能なオペレーティングシステムの核スペースを示す。また図1と同様に、20，22，24は相関コラムであり、夫々が異なるプロセスに対応し、それら夫々のプロセスはそれ自体のユーザ、共用および核スペースを有する。

【0023】図2において、対応するプロセス22のユーザスペース21と共用スペース23を囲む点線26は従来技術の説明のためのものである。従来技術ではそのようなユーザおよび共用スペース21と23はサブルーチンレベルにプロファイルされる。しかしながら、所望されるユーザまたは共用サブルーチンプロファイル毎にそのプロファイルを希望する個々により別々の特定のアクションをとらねばならない。他方本発明ではここに述べるメカニズムは特定の従来のアクションを必要とせずに所望のサブルーチンプロファイルの発生に必要なすべてのデータをとらえる。

10

【0024】図2において、サブプロファイルコラム25が示してある。サブプロファイルは与えられたスペース内の各サブルーチンの規則的なリストである。各サブルーチンについて、そのサブルーチンのアドレス範囲で生じたプログラムカウンタサンプルの総数が与えられる。その目的はプロセス18～24の種々のスペースの代表的なサブルーチンレベルの実行時間プロファイルを概略的に示すことである。このため、例えばプロファイル30はプロセス22に対応するユーザ、共用、および核スペースのプロファイルを示し、サブプロファイル32はプロセス20に対応する同様のユーザ、共用、核スペースプロファイルに対応し、サブプロファイル34はプロセス18に対応するユーザ、共用、核スペースのプロファイルに対応し、プロファイル36はプロセス24に関連したユーザ、共用、核スペースのプロファイリングに対応する。

【0025】図2ではスペースU_4，S_4およびK_4（例えばプロセス24のユーザ、共用および核実行スペース）に対応するプロセス24に注目してみる。ここに示すメカニズムにより、プロセス24の完全な実行時間プロファイル、すなわちその実行スペースのすべてにおいて前述のプロファイル36がつくられる。図2において、「原始ステートメントプロファイル」と記したコラム27は本発明による原始ステートメントレベルプロファイリングが、例えば36でサブプロファイルされているプロセス24のユーザスペースU_4について実行される。この原始ステートメントレベルプロファイル38は、このプロセス24（あるいは他のそのようなプロセス）のユーザスペースの原始ファイル内のコードの各ラインがこのコードラインについて発生される命令の実行にCPUが費やす時間を表わすカウントで注記される。

【0026】図3，4はここに述べるようにシステムプロファイルを発生するためのコンピュータ化されたプロセスを示す流れ図である。図形エレメントを用いるいくつかの約束（コンベンション）が便宜上この図で用いられている。まず、データエレメントの組（tuple:タプル）がここでは表記<データ1，データ2，…>で示される。次に、タプルの中間テーブルまたはタプルのファイルが図3に四角で示されている。次に、人間が見るに適したリポートまたはプログラムリストが矩形で示され

ている。次に、楕円形が補充データをつくるために変更されたシステム呼出しまたはシステム割込みを示す。図3，4の第5の図形エレメントは円でありこれは中間ファイル、リポートまたはテーブルを発生する処理ステップを示す。各中間ファイル、テーブルまたはリポートの主たる内容は図3，4では一つのタプルで示され、これは図示のタイプの固有のタプル群を表わす。

【0027】更に図3，4はライン44により「走行時間処理」40と「ポスト処理」42の二つの領域に分けてある。走行時間処理40についてはブロック46は後述する図5の付加的詳細を表わす。

【0028】図3の上部からまず走行時間処理40を述べると、初期化ステップ52がプロセスIDとプログラム名からなる現在活性のプロセスのテーブルを発生する。このステップは図5で詳述する。中間ファイル37を発生するに必要なデータはシステムプロセスフォーク（fork）56とエクゼック（exec）58で走行時間処理40中維持される。フォーク56はそのフォークを実行したプロセスと同一の名前で新しいプロセスIDをつくる。エクゼック58は現在活性のプロセスに新しい名前を与える。フォーク58またはエクゼック58により夫々新しい<プロセスID，プログラム名>タプルがつくられると、それが維持されてテーブル37をつくる。

【0029】システムCPUディスパッチャに配置されるインストルーメンテーション（instrumentation）により現在のプロセスIDが維持される。AIXオペレーティングシステムはタスク指名（dispatcher）ソフトウェア機能を有する。走行プロセスのフロー内の割込みに続き、このディスパッチャが呼び出されて適正なプロセスをスタートさせる。この割込みは、（1）入力／出力要求による走行プロセスのブロッキング、（2）CPUスケジューリング量子の終了、または（3）外部割込信号によるものである。プロセスのディスパッチ時にディスパッチャコード54がAIX追跡フックを実行し新しく開始されたプロセスのプロセスIDをとらえる。更に、周期的システム割込み（走行時間処理コンポーネント40に示されるプロファイル50）に置かれるインストルーメンテーションにより、タプル<スペースIDアドレス>で示されるプログラムカウンタの値がとらえられる。更に、システムプロセスディスパッチャ（走行時間処理コンポーネント40内に示されるプロファイル54）に置かれたインストルーメンテーションにより、現在のプロセスIDが維持される。図4の処理プロセスはタプル<現在プロセスID>と<スペースID，アドレス>とを組合せて<プロセスID，スペースID，アドレス，カウント>からなる中間ファイル60をつくるに必要なデータを維持する。このタプル内の「カウント」は<プロセスID，スペースID，アドレス>の発生回数であり、プログラムカウンタが“プロセスID”で示されるプロセスについて“スペースID”内の“アドレス”でサンプリングされる回数を示す。走行時間プロファイリングインターバル40の完了時に図3の中間ファイル60と37が書込まれる。

【0030】図3において、本発明のポスト処理42の部分をみるに、処理ステップ39は中間テーブルまたはファイル60と37と共にマージして<プロセスID，スペースID，プログラム名，アドレス，カウント>タプルのテーブル41を得る。リポート31がつくられ、それがシステム内でつかわれるCPUの時間を表わすカウントを加算する。そのようなカウントは各<プロセスID，スペースID，およびプログラム名>についてリポートされる。リポート31は処理ステップ29でつくられ、そしてこれは関連するすべてのアドレス値について固有の<プロセスID，スペースID，プログラム名>タプルに関連したすべてのカウントを加算することにより中間ファイル41を含むタプルを合体させる。

【0031】更に図3において、一つのプログラムのすべてのサブプログラムについてのカウントをリポートするリポート55をつくるに必要な処理ステップが記述される。処理ステップ45はプロファイリング用に選ばれたプログラムエクゼキュータブル43を検査するために用いられて各エクゼキュータブル内のサブプログラムエレメントのアドレス境界を決定する。このステップはUNIX型オペレーティングシステムで周知の“nm”コマンドのようなものを利用するオペレーティングシステムの使用、並びに動的にロードされる共用ライブラリおよび核拡張のアドレススペースを解決するためにオペレーティングシステム状態をポスト処理する、書込まれた特殊なプログラムの使用により行われる。これら特殊なプログラムはAIXオペレーティングシステムの、動的にロードされたモジュールの位置を含むメモリ領域をアクセスする。

【0032】夫々のそのようなプログラムエクゼキュータブルはそれをプログラム名およびスペースIDと関連づけている。二つのプログラムエクゼキュータブル内の各サブプログラムについてスタートアドレスとエンドアドレスがある。処理プロセス45はエクゼキュータブルを検査して各エクゼキュータブルについて、中間ファイル47を形成するタプル<スペースID，プログラム名，サブプログラム名，サブプログラム開始アドレス，サブプログラムエンドアドレス>を得る。

【0033】前記の中間ファイル41内の各タプルに対応するサブプログラム名と相対アドレスを付すことからなる処理ステップ49が次に行われる。この相対アドレスはサブプログラムのスタートアドレスからの変位であり、そして中間ファイル41からのプログラムアドレスおよび中間ファイル47からのサブプログラムスタートアドレスから計算される。結果としての中間ファイル51は<プロセスID，スペースID，プログラム名，相対アドレス，カウント>からなる。

【００３４】リポート５５のサマリがサブプログラムにより使用されるＣＰＵ時間の量を表わすカウントからなるプロセス中に発生され、そして処理ステップ５３により各<プロセスＩＤ，スペースＩＤ，プログラム名>タプルについて発生される。この処理ステップ５３は“相対アドレス”のすべての値にわたりタプル<プロセスＩＤ，スペースＩＤ，プログラム名，サブプログラム名，相対アドレス，カウント>についてカウントを加算する。

【００３５】次に、本発明による詳細プロファイリングについて選ばれたプログラムの注記をもつ原始コードリストを得るに必要な処理ステップを述べる。注記付き原始コードリストは各原始コードラインにより発生される機械命令の実行で使用されるＣＰＵ時間を表わすカウントをリポートするプログラム原始コードリストである。中間ステップとして、詳細プロファイリングについて選ばれた各プログラム用の注記付きアッセンブリコードリストがまず作られる。図４に示すように、プログラムのアッセンブリコードリスト６１を得るためのプログラム原始リスト５７のコンパイレーションである処理ステップ５９が行われる。一つのプログラムの原始コードリストはタプル<プログラム名，サブプログラム名，コード番号の原始ライン，コードテキストの原始ライン>で表わされる。このアッセンブリコードリストはタプル<プログラム名，サブプログラム名，コード番号の原始ライン，コード番号のアッセンブリライン，相対アドレス，コードテキストのアッセンブリライン>で表わされる。

【００３６】エレメント<プログラム名，サブプログラム名，相対アドレス>についての中間ファイル５１のタプルに合致する中間ファイル６１のタプルについては、処理ステップ６３は、リポート６５を含む<プログラム名，サブプログラム名，原始コード番号のライン，アッセンブリコード番号のライン，相対アドレス，コードテキストのアッセンブリライン，カウント>からなる注記付アッセンブリリストを形成するためにデータ<原始コード番号のライン，カウント>を付加する。注記付原始コードリストがステップ６９において同一の<プログラム名，原始コード番号のライン>に対応する中間ファイル６５内のすべてのカウントを加算しそしてその和をプログラム５７の原始リストに付加することで得られる。この注記付原始リスト７１はタプル<プログラム名，サブプログラム名，コード番号ライン，コードテキストライン，カウント>からなる。

【００３７】図５において、本発明のプロファイリングの実時間部分を発生する際用いられるステップが更に詳細に示されている。手順１０４はユーザがプロファイラを呼び出しそしてプロファイリング測定インターバルを特定するとき開始する。この処理ステップはオペレーティングシステムの追跡ファシリティプロセス１００を呼び出す。追跡ファシリティプロセス１００がオンとなると、すべてのＡＩＸ追跡ファシリティフックの特定の副部分が活性化される。これらフックがとらえられるべきイベントの対応する群を生じさせる。以下ではフックとそのフックのイネーブリングにより生じる追跡イベントの両方に同一参照番号を用いる。プロファイル５０、ディスパッチ５４、フォーク５６、初期化５２およびエクゼック５８についての呼出しまたは割込みは追跡ファシリティプロセス１００で特定される。

【００３８】初期化ステップ５２はすべての活性プロセスのプロセス名とＰＩＤをとらえる。これら値は図５のテーブル１０６に置かれる。これら初期値はＡＩＸ“ＰＳ”コマンドを用いて得られる。

【００３９】追跡ファシリティプロセス１００が活性化された後、実時間追跡プロセッサの処理ステップ１０４が待機状態となる。次に追跡ファシリティ処理ステップ１００が追跡フックをそれが生じたときその追跡バッファ１０２に集める。この追跡バッファは二つのバッファに構成するとよい。第１のバッファがフルとなると、以降のイベントは第２バッファに置かれ、そして追跡ファシリティは周知の標準オペレーティングシステムファシリティを用いて実時間追跡プロセッサ処理ステップ１０４を再活性化し、そしてこれが追跡バッファ１０２の内容を処理する。

【００４０】この処理は追跡バッファ１０２に記憶された逐次的追跡イベントをステップしそして夫々のイベントのタイプにつき後述するような適正なアクションを行うことから成る。

【００４１】初期化ステップ５２は追跡ファシリティ１００がオンとなるとき活性である各プロセスについてＰＳコマンドを用いてＰＩＤ、プロセス名タプルをレコードする。実行(exec)フック５８は新しいプロセス名を含む。このフックにより、１０６に実行フックから現在ＰＩＤと新プロセス名とからなる新しいエントリがつくられる。フォークフック５６は新しくつくられるプロセスのＰＩＤを含む。このフックにより新ＰＩＤと現プロセス名からなるタプルが１０６に加えられる。

【００４２】ディスパッチフック５４は新しくディスパッチされるプロセスのＰＩＤを含む。プロファイルフック５０はＳＩＤとそのスペース内の相対アドレスのタプルを含む。このフックの処理はすべてのプロセスと合理的なメモリスペース内のすべてのスペースをプロファイルするに必要なすべての情報を維持する機能をサポートする際重要である。使用されるこの技術はハッシュテーブルとして周知のものである。このテーブルへのキーはＰＩＤ（現プロセスＩＤ）とプロファイルフック１５０からのＳＩＤとアドレスの関数である。キーが整合した（ＰＩＤ，ＳＩＤ，およびアドレス）ハッシュスロットが見つかれば１０８の関連カウントフィールドが増加する。この処理はプロファイリングの実時間相において連続する。プロファイルプロセスが終了すると、実時間追

15

跡プロセッサ機能１０４が追跡ファシリティプロセス１００を減勢し、そしてこれにより残りの追跡バッファの内容が１０４に移されて処理される。この機能が完了すると、テーブル１０６と１０８が前述のポスト処理用にファイル３７と６０（図３，４）に夫々書込まれる。

【００４３】図６は以上のプロファイリングシステムに適した代表的なコンピュータシステムを示す。中央処理ユニット１２２が設けられ、これはプログラムカウンタ１２４を含み、このカウンタは１２８で示す形のアドレス１３２を含む。一つのアドレスが仮想メモリ１２０にスペースＩＤ１３６として置かれ、その後にそのスペースＩＤ内の変位アドレスが続く。例えば、１３０で示すスペースＩＤ２と変位アドレス１００は仮想メモリの第３のスペースＩＤブロックをポイントし、命令ワードはブロック１４２内に１４４で示す相対アドレス１００をポイントする。

【００４４】ＩＢＭのＲＩＳＣシステム／６０００のような本発明によるプロファイリングに適したエンバイロメントの一実施例では１３８で示すこの仮想メモリ１４６は$2^{24}-1$個の同一構成のメモリブロックで形成され、スペースＩＤ１３６が１３８で示すようにインデクス$2^{24}-1$を参照する。変位アドレスは１３４で示すように０から$2^{28}-1$の範囲であり、図６のメモリシステム内のメモリエレメントの数は2^{28}となる。例えば、スペースＩＤ、アドレスタプル１３０は前述のメモリワード１４４で表わされるスペースＩＤ＝２および変位アドレス＝１００の内容を限定する。プログラムカウンタ１２４の目的はコンピュータプログラム内の命令をステップすることであり、プログラムカウンタ１２４の内容はスペースＩＤとワード１４４のような特定のメモリワード

16

の相対アドレスである。プログラムカウンタ１２４のすべての値について、メモリワードの内容は、ＣＰＵ１２２の第２コンポーネントを含む命令ユニット１２６に自動的にコピーされる。

【図面の簡単な説明】

【図１】本発明によりつくられる総合プロファイルの要約を示す概略図。

【図２】多重プロセス、多重スペース計算エンバイロメントと本発明のプロファイリングプロセスとの間の関係を示す図。

【図３】本発明のプロファイリングプロセスのフロー図の一部。

【図４】本発明のプロファイリングプロセスのフロー図の一部。

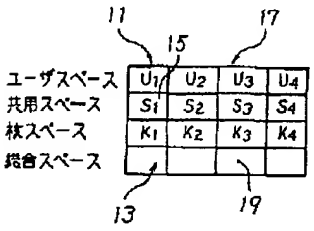
【図５】図３，４の実時間プロファイラの処理を詳細に示す機能ブロック図。

【図６】本発明のプロファイリングシステムおよび方法が動作する代表的なコンピュータシステムエンバイロメントのブロック図。

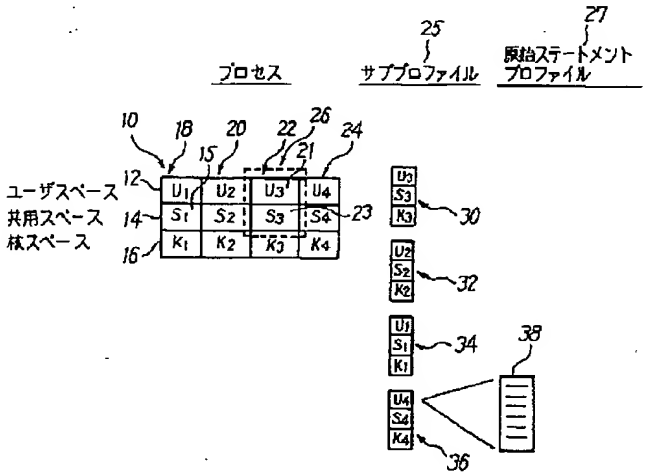
【符号の説明】

５４　ディスパッチ
５０　プロファイル
５６　フォーク
５２　初期化ステップ
５８　エクゼック（実行）
１００　追跡ファシリティ
１０２　追跡バッファ
１０４　実時間追跡プロセッサ
１０６　プロセス状態
１０５　アドレステーブル

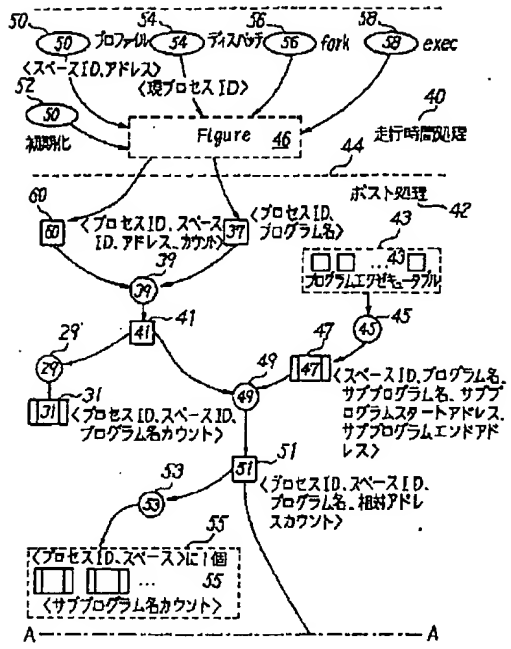
【図１】



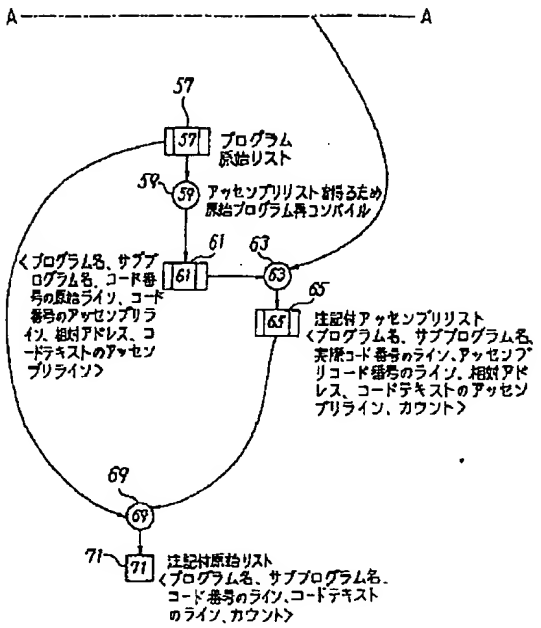
【図２】



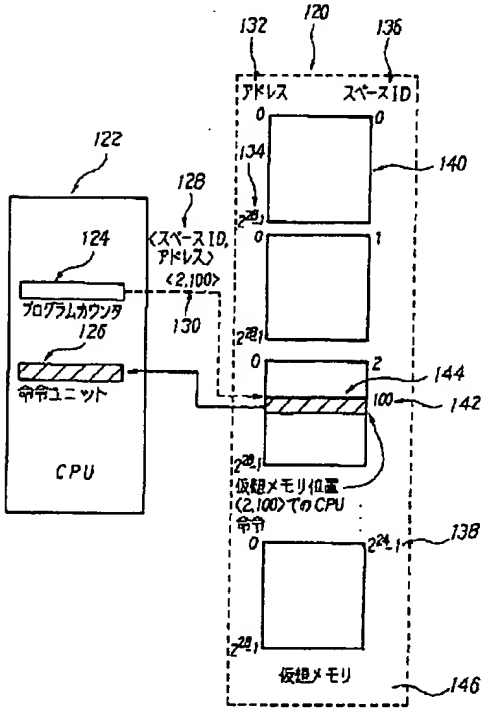
【図3】


50
プロファイル
54
ディスパッチ
56
fork
58
exec
〈スペースID,アドレス〉
〈現プロセスID〉
52
初期化
Figure 46
40
走行時間処理
44
ポスト処理
42
60
〈プロセスID,スペースID,アドレス,カウント〉
37
〈プロセスID,プログラム名〉
43
プログラムエグゼキュータブル
39
41
45
29
47
49
〈スペースID,プログラム名,サブプログラム名,サブプログラムスタートアドレス,サブプログラムエンドアドレス〉
31
〈プロセスID,スペースID,プログラム名カウント〉
51
53
〈プロセスID,スペースID,プログラム名,相対アドレスカウント〉
55
〈プロセスID,スペース〉に1個
〈サブプログラム名カウント〉
A
A

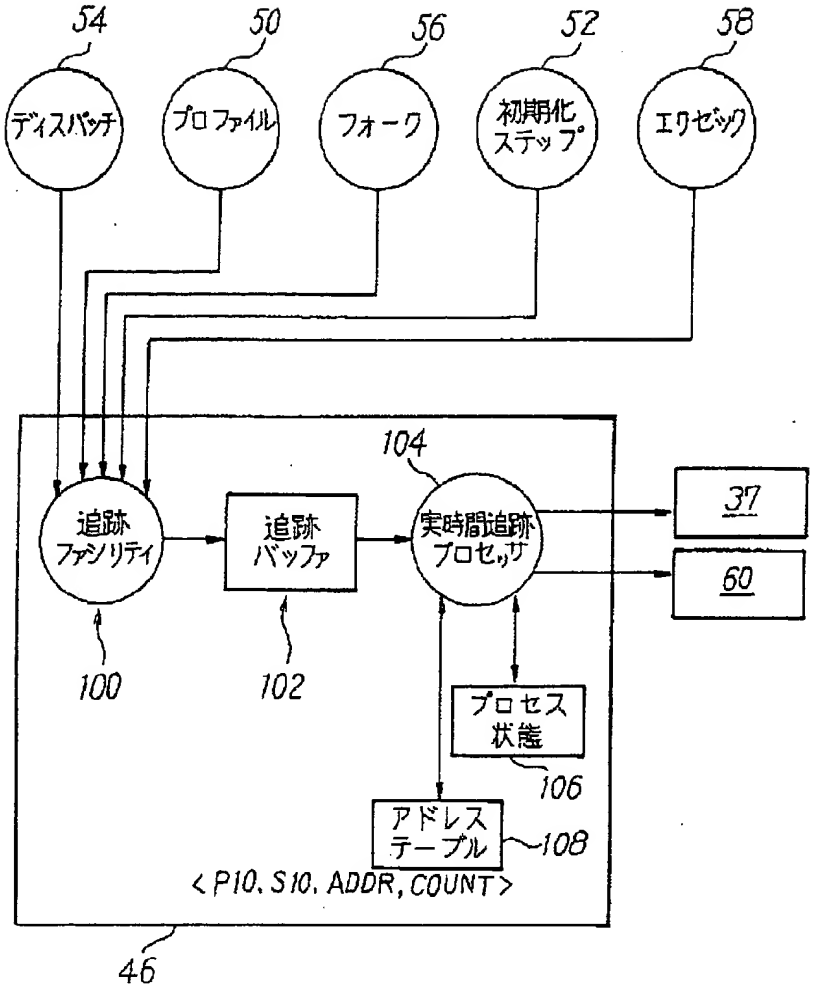
【図4】


A
A
57
プログラム原始リスト
59
アッセンブリリスト得るため原始プログラム再コンパイル
〈プログラム名、サブプログラム名、コード番号の原始ライン、コード番号のアッセンブリライン、相対アドレス、コードテキストのアッセンブリライン〉
61
63
65
注記付アッセンブリリスト
〈プログラム名、サブプログラム名、実際コード番号のライン、アッセンブリコード番号のライン、相対アドレス、コードテキストのアッセンブリライン、カウント〉
69
71
注記付原始リスト
〈プログラム名、サブプログラム名、コード番号のライン、コードテキストのライン、カウント〉

【図6】


132
120
136
アドレス
スペースID
122
128
124
〈スペースID,アドレス〉
〈2,100〉
プログラムカウンタ
130
126
命令ユニット
CPU
134
140
144
142
100
仮想メモリ位置〈2,100〉でのCPU命令
138
仮想メモリ
146

【図5】



実時間プロファイラ処理

フロントページの続き

(72)発明者　ロバート、ジョン、アーカート
アメリカ合衆国テキサス州、オースチン、
マイスティック、オークス、トレイル、
9210